

Temporal Action Localization by Structured Maximal Sums

Zehuan Yuan^{1,2}, Jonathan C. Stroud², Tong Lu¹, and Jia Deng²

¹State Key Laboratory for Novel Software Technology, Nanjing University, China

²University of Michigan, Ann Arbor

Abstract

We address the problem of temporal action localization in videos. We pose action localization as a structured prediction over arbitrary-length temporal windows, where each window is scored as the sum of frame-wise classification scores. Additionally, our model classifies the start, middle, and end of each action as separate components, allowing our system to explicitly model each action’s temporal evolution and take advantage of informative temporal dependencies present in this structure. In this framework, we localize actions by searching for the structured maximal sum, a problem for which we develop a novel, provably-efficient algorithmic solution. The frame-wise classification scores are computed using features from a deep Convolutional Neural Network (CNN), which are trained end-to-end to directly optimize for a novel structured objective. We evaluate our system on the THUMOS ’14 action detection benchmark and achieve competitive performance.

1. Introduction

In temporal action localization, we are given a video and aim to detect *if* and *when* a particular action occurs. Specifically, we answer three questions – “*is there an action in the video?*”, “*when does the action start?*”, and “*when does the action end?*”. By automating this process, we can enable people to efficiently search through the millions of hours of video data which are generated each and every day. However, this remains a challenging problem for several reasons. Crucially, actions have inherent temporal structure, so we require a representation that can model the *temporal evolution* of actions in addition to their instantaneous spatial appearance. Previous methods have either failed to model temporal evolution, or done so at significant computational cost [11, 16, 15]. High computational cost is a significant problem for these methods, because in many

This work was done while Zehuan Yuan was a visiting student at the University of Michigan.

practical applications, the videos of interest may be arbitrarily long, and methods must gracefully scale to videos that last hours (*e.g.* movies, web videos) or even days (*e.g.* security footage, first-person vision). Finally, extracting powerful features for detecting actions in videos remains an unsolved challenge.

To overcome these challenges, we propose a method that directly models the temporal evolution of actions, and we develop a provably-efficient algorithm to perform localization in this framework. Our temporal evolution framework is based on the observation that all actions have a *start*, *middle*, and *end*, and that these components each have distinct patterns of appearance and motion. We hypothesize that by localizing these three action parts separately, we can significantly improve localization performance by enforcing consistent structure in their ordering. Specifically, we model an action as a *temporal window* – a variable-length sequence of video frames – and we assume that each temporal window begins with a single *start* frame, followed by one or more *middle* frames, and finally a single *end* frame (Figure 1). We otherwise impose no restrictive constraints on the temporal sequence of each action. In doing this, we recover just enough temporal information to take advantage of the inherent structure present in each action, without requiring any additional annotations or making unrealistic assumptions about the composition of actions.

At test time, we localize actions by searching for the *structured maximal sum* – the sequence of start, middle, and end frames which has the highest sum of corresponding frame-wise confidence scores. Solving this problem is non-trivial, as it naïvely requires a search over a quadratic number of possible start-end pairs. However, in Section 4, we propose a novel dynamic programming algorithm which provably finds the top- k structured maximal sums for a video of arbitrary length. We prove that this algorithm is efficient, and specifically we show that it finds the structured maximal sum in linear time. Our solution is related to that of the well-studied *k-maximal sums problem*, for which similar efficient algorithms exist [2]. Our structured maxi-

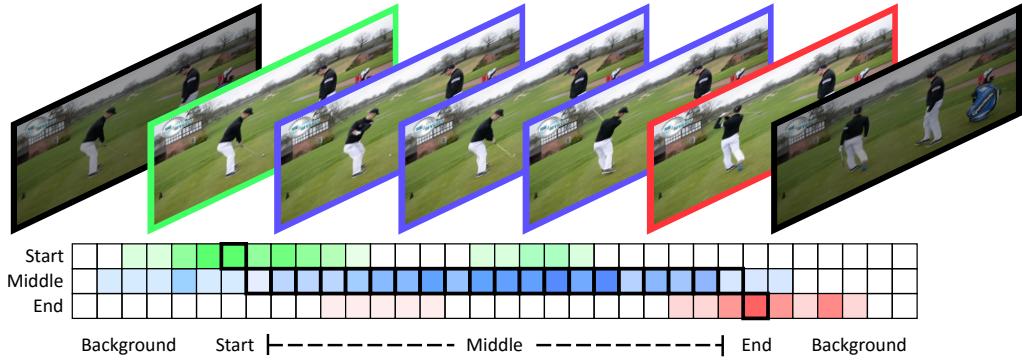


Figure 1: Temporal evolution of a *golf-swing* action. Our system explicitly models evolution as a single *start* frame (green), followed by many *middle* frames (blue), and a single *end* frame (red).

mal sum algorithm enables us to gracefully scale localization to arbitrary-length untrimmed videos, while simultaneously encoding the temporal evolution of each action.

We classify the three action components separately using powerful discriminative features from two-stream Convolutional Neural Networks (CNNs) [16]. In Section 5.2, we train the entire system end-to-end, using a novel structured loss function. We train and evaluate our approach on the THUMOS’14 challenge dataset [6] in Section 6 and achieve competitive results.

Our primary contribution is a framework that allows us to model the temporal evolution of actions without sacrificing efficient temporal localization. Crucial to our framework is a novel, provably-efficient algorithm, which computes the structured maximal sum in linear time. We achieve competitive results on action detection baselines, and present a number of ablation studies to demonstrate the contributions of each component of our pipeline.

2. Related work

Temporal action localization in videos is an active area of research, and much recent progress has been facilitated by an abundance of benchmark datasets and competitions which focus on temporal localization, including the THUMOS [6] and ActivityNet [7] challenges. Most prior approaches have fallen into one of two categories: *sliding window* and *framewise* classification. In this section we will outline the major contributions of these approaches.

Sliding Window. Many leading approaches for temporal localization apply classifiers to fixed-width windows that are scanned across each video. These approaches have the advantage that they are able to consider contextual information and temporal dependencies in their classifications, up to the size of the temporal windows. Oneata *et al.* [11], the winners of the THUMOS ’14 localization competition, used sliding window classifiers applied to fisher vector representations of improved dense trajectories features [25]. Wang

et al. took a similar sliding-window approach and came in second place in the same competition [26]. Recently, Shou *et al.* proposed a sliding 3D Convolutional Neural Network for localization, in favor of the hand-designed features of previous methods [15]. In a similar vein to our temporal evolution model, Gaidon *et al.* used sliding window classifiers to locate action parts (actoms) from hand-crafted features [4]. The key distinction between their sequence model and ours is that their action parts are specific to each individual action, and must be chosen and labeled manually, while ours uses the same parts for each action, and requires no additional annotations.

Most sliding window approaches are applied at multiple window sizes to account for variation in temporal scale. This leads to significant redundant computation and makes these methods expensive to scale to long videos. However, their success in competitions like THUMOS demonstrate that the contextual information afforded by sliding window methods is important for accurate localization.

Frame-wise Classification. Another class of popular approaches apply classifiers to each individual frame to detect the presence or absence of an action. Action windows are then aggregated during post-processing, using simple non-maximum suppression or more complex sequence models. Singh *et al.* [19] achieved competitive performance in the ActivityNet challenge [7] using a frame-wise classifier to propose action locations, aggregated together by minimizing a loss that encourages piecewise smoothness in the detections [19]. Sun *et al.* and Singh *et al.* applied frame-wise CNN feature detectors, connected by recurrent LSTM modules [21, 18]. Richard *et al.* adopt language models applied to traditional motion features [12].

While each of these methods are able to incorporate some temporal context in post-processing, they each either rely on hand-designed frame-level features or optimize some frame-level loss. Our approach, by contrast, is trained end-to-end, and directly optimizes a structured loss over temporal action windows, allowing it to learn features that

facilitate accurate action localization.

Other Approaches. Max-margin losses have been used to detect actions in an online setting [8] and from 3D video features [28], but not in an end-to-end pipeline. Many works have taken other approaches to model the temporal structure present in actions [22]. Recently, Yeung *et al.* [29] proposed using reinforcement learning to actively search for informative frames in a video before directly regressing the start and end points of each action. Their approach is efficient, in that it only needs to observe a few frames before making each prediction, but it does not aggregate information over the entire video to achieve the best performance.

The work that is perhaps most related to ours is [24], which approaches action localization as a structured prediction over spatio-temporal paths through a video, utilizing the max-path algorithm of [23] to perform efficient inference. Their method is capable of performing both spatial and temporal localization jointly, and similarly uses a max-margin structured regression to learn frame-wise classification scores. Our method, however, has the advantage of modeling the temporal evolution of actions, and utilizes powerful CNN features, which we train end-to-end.

3. Localization as Structured Prediction

Suppose we are given a video $v = \{x_1, x_2, \dots, x_n\} \in \mathcal{V}$, where x_t denotes the frame at timestep t , and n is the total number of frames in the video. We define a *temporal window* to be a contiguous segment of video frames $y = \{x_s, x_{s+1}, \dots, x_e\} \in \mathcal{Y}$, where s and e are the indices of the start and end frames, respectively, and $1 \leq s \leq e \leq n$. Furthermore, suppose that each frame has a real-valued *frame-wise score* $f(x_t) \in \mathbb{R}$ which can be positive or negative, which indicates our confidence in frame x belonging to an instance of a particular action class. Note that, for convenience, we express $f(x_t)$ as a function of only a single frame x_t , while in practice f may depend on features extracted from the entire video. For a video and corresponding temporal window, we define the *confidence score* $F : \mathcal{V} \times \mathcal{Y} \mapsto \mathbb{R}$ as the sum of framewise scores, that is, $F(v, y) = \sum_{t=s}^e f(x_t)$. The predicted temporal window for video v is the one that maximizes the confidence score, in particular, $\hat{y} = \arg \max_{y \in \mathcal{Y}} F(v, y)$.

Naïvely, by searching over all possible start- and endpoint pairs, this maximization requires a search over a space quadratic in the number of frames. For long videos, this is impractical. However, because F is decomposable into frame-wise scores, we can pose this as the classic *maximum sum* problem [2], for which there exists an $\mathcal{O}(n)$ -time solution [1]. In practical settings, we may have multiple action instances in a single video. Finding the k -best windows can similarly be posed as a k -maximal sums problem, for which there exists a $\mathcal{O}(n + k)$ -time solution [3]. In the following section, we model more complex temporal dependencies.

3.1. Temporal Evolution Model

Actions vary greatly in their appearance and motion characteristics over time. By explicitly modelling the *temporal evolution* of an action, we can take advantage of this inherent temporal structure. In particular, we notice that the frames at the start- and end-points of an action instance tend to vary greatly in appearance, as the actor will often change position (as in *basketball-dunk*) or pose (as in *golf-swing*) over the course of the action. Additionally, frames in the middle of the action instance tend to have different motion characteristics than the start- and end-points as the actor performs complex body movements.

The start and end of an action are of particular importance in temporal localization, as they define the boundaries of a single action instance. In order to encourage precise localization, we explicitly model each action as a single *start* frame, followed by an arbitrary-length series of *middle* frames, and finally a single *end* frame. Suppose we have separate signed frame-wise confidence scores $f^s(x)$, $f^m(x)$, and $f^e(x)$, for the *start*, *middle*, and *end* components, respectively. Using this new formulation, we can rewrite the confidence score $F(v, y)$ for a video v as

$$F(v, y) = \lambda_s f^s(x_s) + \lambda_m \sum_{t=s+1}^{e-1} f^m(x_t) + \lambda_e f^e(x_e) \quad (1)$$

where λ_s , λ_m and λ_e are parameters that specify the relative importance of each action part. In our experiments, we set $\lambda_s = \lambda_m = \lambda_e = 1$ except where stated otherwise.

This generalization comes with a number of advantages over the single-class confidence score without temporal evolution. First, we are penalized heavily for detections that fail to find good matchings for the start and end frames. This enforces temporal consistency, as the best detections will be those that successfully match each of the three components in their correct order. This resistance to illogical matchings gives us robustness to variance in the frame-wise scores. This makes us less likely to merge consecutive or partial instances of an action into a single detection, and encourages the detector to stretch each detection to the full extent of the action instance, preventing over- and under-segmentation. Finally, start and end labels are readily available from existing temporal action annotations, meaning that we require no additional training data. Finally, since every action has a *start*, *middle*, and *end*, this formulation makes no restrictive assumptions about the structure of complex actions.

4. Structured Maximal Sums

Given frame-wise scores f^s , f^m , and f^e , and a video v , our goal is to detect all instances of a particular action. We represent these detections as the top- k temporal windows, as ordered by their confidence scores in Equation 1.

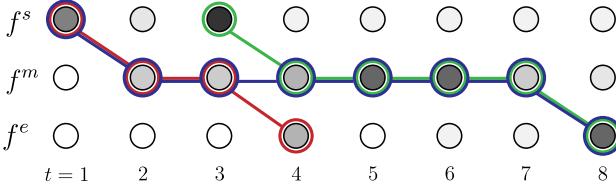


Figure 2: A depiction of the structured maximal sums problem. Gray circles depict classification scores for each of the three action components (darker is higher), and colored outlines depict plausible temporal localizations. The three windows depicted here are red ($t = 1 \dots 4$), blue ($t = 1 \dots 8$), and green ($t = 3 \dots 8$)

In Section 3, we showed how, without temporal evolution, localization can be framed as a k -maximal sums problem [1]. However, our formulation introduces additional challenges, as we now need to compute the top- k *structured* maximal sums (Figure 2), which previous work does not address. To solve this problem, we introduce the Structured Maximal Sums (SMS) algorithm (Algorithm 1), which efficiently finds the top- k structured maximal sums.

The Structured Maximal Sums algorithm makes a single pass through the video, maintaining the value of the K -best windows found so far in the list $\text{kmax}[:]$, which is kept in sorted order. It also keeps track of the values of the K -best *incomplete temporal windows* that end at frame i in $\text{rmax}[:]$, that is, the windows that end at i but do not include an endpoint $f^e[i]$. We now prove the correctness of the SMS algorithm.

For clarity, we first introduce the following notation. We assume that all frame-wise classifier scores are pre-computed, and are contained in ordered lists $f^s[1 \dots n]$, $f^m[1 \dots n]$, and $f^e[1 \dots n]$, and the shorthand $[:]$ refers to all elements in a list simultaneously. Similarly, we denote adding a value to each member of a list as $f[:] + n$. We denote the operation of inserting an item s into a sorted list kmax as $\text{merge}(s, \text{kmax})$. We denote the k -th maximum value of a function g over a discrete space \mathcal{X} as $\text{k-max}_{x \in \mathcal{X}} g(x)$.

Lemma 1. *Let $\text{rmax}_i[:]$ denote the list of K -best incomplete temporal windows ending at timestep i , not including the end-point $f^e[i]$. Namely, let*

$$\text{rmax}_i[k] = \underset{j \in \{1, \dots, i\}}{\text{k-max}} \left\{ f^s(j) + \sum_{q=j+1}^i f^m(q) \right\}. \quad (2)$$

Then $\text{merge}(f^s[i+1], \text{rmax}_i[:] + f^m[i+1])$ gives the list of the K -best incomplete temporal windows ending at timestep $i+1$.

Proof. The k -th best incomplete window ending at frame $i+1$ is either the window that starts at $i+1$, or it is a continuation of one of the K -best windows that ended at frame i .

Algorithm 1 Top- K Structured Maximal Sums

Input: Frame-wise scores $f^s[1 \dots n]$, $f^m[:]$, and $f^e[:]$
Output: $\text{kmax}[1 \dots K]$

```

for each  $k \leftarrow 1$  to  $K$  do
     $\text{kmax}[k] \leftarrow -\infty$ ,  $\text{rmax}[k] \leftarrow -\infty$ 
end for
 $\text{rmax}[1] \leftarrow f^s[1]$  {Initialization}
for each  $i \leftarrow 2$  to  $n$  do
    for each  $k \leftarrow 1$  to  $K$  do
         $s \leftarrow \text{rmax}[k] + f^e[i]$ 
         $\text{rmax}[k] = \text{rmax}[k] + f^m[i]$ 
         $\text{kmax}[:] = \text{merge}(s, \text{kmax}[:])$ 
    end for
     $\text{rmax}[:] = \text{merge}(f^s[i], \text{rmax}[:])$ 
end for

```

$\text{rmax}_i[:] + f^m[i+1]$ gives the list of all continuations of the previous K -best incomplete windows. We insert $f^s[i+1]$ to this list, and discard at most one of the K continuations if $f^s[i+1]$ is greater than it. What remains are the K -best incomplete temporal windows ending at frame $i+1$. \square

Lemma 2. *Let $\text{kmax}_i[:]$ denote the list of the K best temporal windows ending at or before frame i . Then $\text{merge}(\text{rmax}_i[:] + f^e[i+1], \text{kmax}_i[:])$ gives the list of the K -best temporal windows ending at timestep $i+1$.*

Proof. We know from Lemma 1 that $\text{rmax}_i[:]$ gives the K -best *incomplete* temporal windows ending at frame i . The k -th best temporal window ending at frame $i+1$ is either one of these incomplete windows, completed by adding $f^e[i+1]$, or it is one of the top complete windows already contained in kmax_i . By merging these two lists, we select the top- K windows overall, preserving the top- K complete temporal windows. \square

Each rmax_i and kmax_i (including the call to merge) can be constructed in $\mathcal{O}(K)$ time [1]. We compute kmax_i for all $i \in \{1, \dots, n\}$, so the total time complexity is $\mathcal{O}(nK)$. This result, and the results of the above lemmas, lead us to our primary theoretical contribution:

Theorem 4.1. *The SMS algorithm computes the K -best temporal windows in a video of length n in $\mathcal{O}(nK)$ time.*

We note that, while this algorithm as written only computes the *scores* of the top- K temporal windows, our implementation is able to recover the windows themselves. This is accomplished with simple bookkeeping which keeps track of the temporal windows' start- and end-points as they are added to the rmax and kmax lists.

5. Training

So far, we have assumed that all frame-wise action scores f^s , f^m , and f^e are computed beforehand. In this sec-

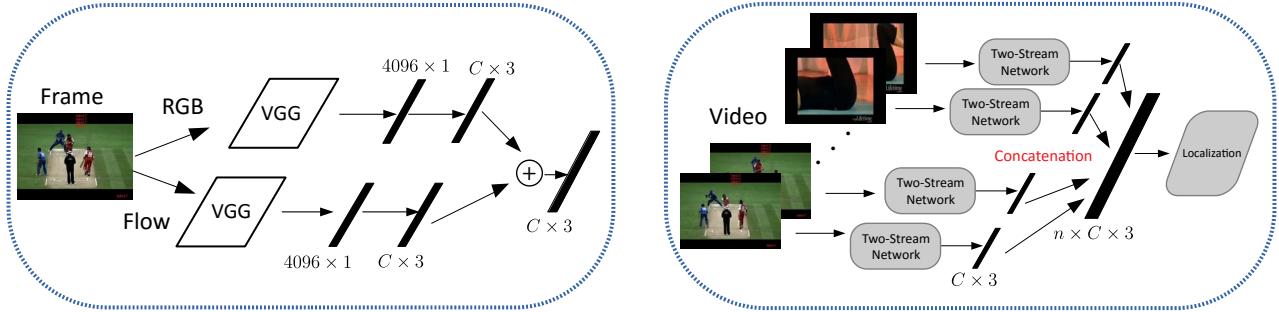


Figure 3: Diagram of our localization architecture. (left) We use a two-stream network with a VGG backbone architecture to generate frame-wise confidence scores. (right) Scores from n frames are concatenated and localization is performed.

tion, we describe how these frame-wise score functions are learned. We use deep Convolutional Neural Networks (CNNs) to produce the score functions, and introduce a structured loss function which can be used to train these CNNs in an end-to-end framework.

5.1. Network Architecture

We adopt the two-stream network architecture of [16] to extract deep spatio-temporal features for each video frame. The two-stream architecture consists of two Convolutional Neural Networks (CNNs), *Spatial-CNN* and *Motion-CNN*. The first network stream, *Spatial-CNN*, operates on the color channels of a single video frame, capturing features from the static appearance of the scene. The second stream, *Motion-CNN*, operates on dense optical flow fields computed between adjacent frames, capturing distinctive motion features and pixel trajectories over time. However, unlike [16], we adopt the larger VGG 15-layer network from [17] as the backbone architecture for each of the two streams. For each stream, we produce $(C \times 3)$ -dimensional outputs, where C is the total number of action classes. Finally, we average the frame-wise scores from the two streams and concatenate the results across frames. This architecture is pictured in Figure 3.

5.2. Structured Loss

Our model minimizes a video-level structured loss function, rather than the frame-wise loss function used to train the typical two-stream action recognition architecture. By directly optimizing for temporal action localization, we enable the frame-wise scores to take into account the temporal evolution of actions. This enables a level of fine-tuning that would not be possible by optimizing for a frame-wise objective. As in the typical two-stream architecture, we first pre-train each stream separately and fuse by finetuning.

We have a dataset $V = \{v_1, v_2, \dots, v_m\}$ of m training videos and labels $Y = \{y_1, y_2, \dots, y_m\}$. Each video $v_i = \{x_1^{(i)}, \dots, x_{n_i}^{(i)}\} \in \mathcal{V}$ can be arbitrarily long, and its length is denoted n_i . For simplicity, we assume the training

videos contain only a single action instance. These labels $y_i = (s^{(i)}, e^{(i)}, \ell^{(i)}) \in \mathcal{Y}$ consist of a start index s , end index e , and action label ℓ . Our goal is to learn a confidence function $F : \mathcal{V} \times \mathcal{Y} \mapsto \mathbb{R}$, which measures how likely it is that a particular action instance is present in the video. We require F to take on the frame-wise summation form as in Equation 1. We denote the learnable parameters of F , namely those of the CNNs, as \mathbf{w} . We use the notation $F(v, y; \mathbf{w})$ to denote the confidence score produced for a video v and window y with parameters \mathbf{w} .

For a training video v_i , we define the localization loss \mathcal{L}_{loc} as the gap between the highest-scoring temporal window and the ground truth label for the action ℓ^i :

$$\mathcal{L}_{loc}(v_i) = \left[\max_{y \neq y_i} \{\Delta(y_i, y) + F(v_i, y; \mathbf{w})\} - F(v_i, y_i; \mathbf{w}) \right]_+, \quad (3)$$

where $[\cdot]_+ = \max(0, \cdot)$ is the hinge loss function [5]. The Δ term is added to weaken the penalty on windows with high overlap with ground truth, and is defined as $\Delta(y, \bar{y}) = |y \cup \bar{y}| - |y \cap \bar{y}|$, where each predicted window y and \bar{y} is a set of video frames and $|\cdot|$ is the cardinality.

To further make the network more discriminative, we introduce a classification loss \mathcal{L}_{cls} , which enforces that the estimated windows of other actions should have lower scores than those of the ground truth action class. We define

$$\mathcal{L}_{cls}(v_i) = \frac{1}{C-1} \left[M + \max_{y: \ell \neq \ell^i} F(v_i, y; \mathbf{w}) - F(v_i, y_i; \mathbf{w}) \right]_+, \quad (4)$$

where M is a fixed parameter that ensures we do not penalize the detection if the distance is already lower than M . In our experiments, we set M to be the ground truth window length $|y_i|$ of the video v_i .

The full structured objective \mathcal{L} is a sum of the two losses over all videos in the training set, defined as follows:

$$\mathcal{L}(V) = \sum_{i=1}^m (\mathcal{L}_{loc}(v_i) + \lambda \mathcal{L}_{cls}(v_i)), \quad (5)$$

where λ weights the relative importance of the two loss functions. By default, we set $\lambda = 0.5$.

Algorithm 2 Loss-Augmented Structured Maximal Sum

Input: Confidence scores $f^s[1 \dots n]$, $f^m[:]$ and $f^e[:]$; ground truth window $y = \{s, e\}$

Output: smax

```

 $\text{smax} \leftarrow -\infty; \text{rsum}[1] \leftarrow -\infty; p \leftarrow 0$  {Initialization}
for each  $i \in [1, s) \cup (e, n]$  do
     $f^s[i] \leftarrow f^s[i] + 1; f^m[i] \leftarrow f^m[i] + 1; f^e[i] \leftarrow f^e[i] + 1;$ 
end for
for each  $j \leftarrow 2$  to  $N$  do
     $len \leftarrow \max[0, \min(e - s + 1, e - j)]$ 
     $\text{rsum}[j] \leftarrow \max(\text{rsum}[j - 1] + f^m[j], p + f^s[j])$ 
     $\text{smax} \leftarrow \max(\text{smax}, \text{rsum}[j - 1] + len + f^e[j])$ 
    if  $j \in [s, e]$  then
         $p \leftarrow p + 1$ 
    end if
end for

```

Note that both loss functions are typical structural SVM losses and the parameters \mathbf{w} can therefore be learned in a similar end-to-end fashion [14]. Since both the localization and classification losses are sub-differentiable, the parameters of the two CNN streams can be learned by backpropagation. Specifically, for one layer l , the gradient of either loss on one video $\mathcal{L}_{(\cdot)}(v_i)$ with respect to that layer's parameters, $w^{(l)}$, can be calculated as

$$\frac{\partial \mathcal{L}_{(\cdot)}(v_i)}{\partial \mathbf{w}^{(l)}} = \left(\frac{\partial \mathcal{L}_{(\cdot)}(v_i)}{\partial F(v_i, y_i^*)} \frac{\partial F(v_i, y_i^*)}{\partial \mathbf{f}} \right. \\ \left. - \frac{\partial \mathcal{L}_{(\cdot)}(v_i)}{\partial F(v_i, y_i)} \frac{\partial F(v_i, y_i)}{\partial \mathbf{f}} \right) \frac{\partial \mathbf{f}}{\partial \mathbf{w}^{(l)}} \quad (6)$$

where y_i^* represents $\arg \max_y (\Delta(y, y_i) + F(v_i, y))$ for \mathcal{L}_{loc} and $\arg \max_{y, l \neq \ell^i} F(v_i, y)$ for \mathcal{L}_{cls} . \mathbf{f} is the set of frame-wise confidence scores produced by the neural networks.

The gradient of \mathbf{f} with respect to the network parameters, $\frac{\partial \mathbf{f}}{\partial \mathbf{w}^{(l)}}$, can be computed via backpropagation, as is standard for CNNs without the structured objective. Therefore, it remains to compute two gradients: (1) the gradient of the confidence function F w.r.t. the classifiers \mathbf{f} and (2) the gradient of the objective function $\mathcal{L}_{(\cdot)}$ w.r.t. F . To compute (1), we recall that confidence function is simply the summation of the action parts scores, so its gradient computation is straightforward. Although (2) is not differentiable, it is in fact sub-differentiable, so we compute a subgradient:

$$\frac{\partial \mathcal{L}}{\partial F(v_i, y_i^*)} = \begin{cases} 1 & \text{if } \Delta(y_i, y_i^*) + F(v_i, y_i^*) - F(v_i, y_i) > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

This allows us to train end-to-end with subgradient descent.

To compute the subgradient, we need to find the best window y_i^* . We use the SMS algorithm (Algorithm 1) to

find y_i^* in \mathcal{L}_{cls} . However, because of the Δ term in the maximization in \mathcal{L}_{loc} , in order to compute y_i^* we are required to perform a maximization of the *loss-augmented confidence*. In Algorithm 2, we modify the SMS algorithm to include this term, achieving the same linear time complexity, guaranteeing that this can be computed efficiently during training. Additionally, we only compute the top detection.

6. Experiments

We evaluate our method on the THUMOS'14 dataset [6]. Our implementation is built on Caffe [9].

Two-stream neural networks. The inputs to Spatial-CNN are RGB video frames cropped to 224×224 with the mean RGB value subtracted. The inputs to Motion-CNN are dense optical flow channels computed by the TVL1 optical flow algorithm [30]. We scale each optical flow image to be between [1, 255] and stack the flows of 10 frames in both directions to form a $224 \times 224 \times 20$ 3D volume. Spatial-CNN is pre-trained for object recognition on ImageNet [13], and Motion-CNN is pre-trained for action classification on UCF101 [20]. We train Spatial-CNN and Motion-CNN separately, then jointly fine-tune both of their final two fully-connected layers. Additionally, we adopt multi-scale random cropping for both streams. For each sample, we first randomly choose a scale from a predefined list, then choose a random crop of size $(224 \times 224) \times scale$. The cropped region is resized to 224×224 before being input into the network. For Spatial-CNN, three scales [1, 0.875, 0.75], for Motion-CNN, we use two scales [1, 0.875].

Postprocessing. We divide each testing video into overlapping 20-second snippets with an 18-second overlap between neighboring snippets, and perform localization independently for each snippet. Subsequently, we merge predictions across these snippets. We set the number of action instances to $K = 100$, as experimentally $K \geq 100$ does not improve recall on our validation set. The temporal action window scores from Equation 1 are prone to giving higher scores to longer windows, so we additionally normalize the confidence of each window by its window length. Furthermore, we multiply confidence scores by action duration priors as in [11] to encourage action windows to have reasonable lengths. After generating all candidates, we filter those with large overlap using non-maximum suppression.

Balanced training. Middle frames are more prevalent than start and end frames, so to prevent the network from becoming biased towards middle frames, we divide each middle frame's score by the total window length during training. In addition, since the manual annotations of start and end of actions are relatively noisy, we randomly sample the start frame from the first 10% of frames and the end from the last 10%. Middle frames are sampled from the middle 80%.

Evaluation Metric. We use mean Average Precision (mAP) to measure localization performance as in [6]. We

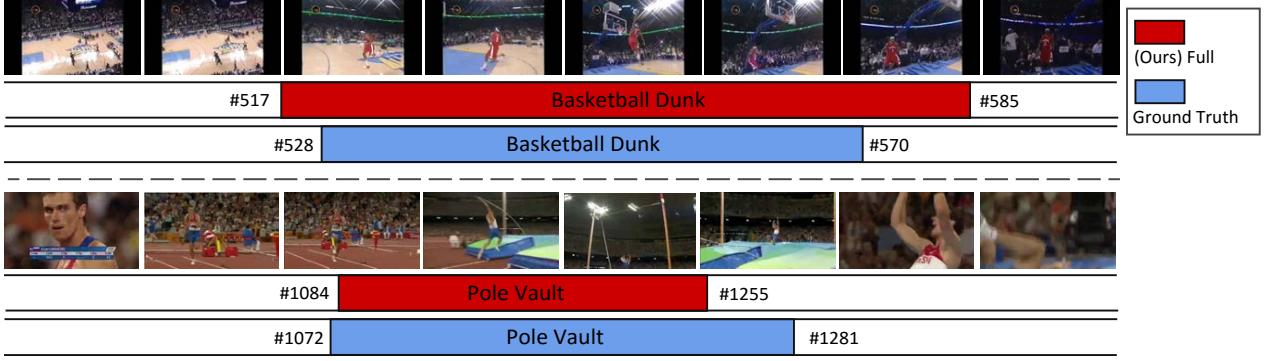


Figure 4: Example detections from our system. Frame numbers are given at the boundaries of each action.

count a detection as correct if it predicts the correct action label and its intersection over union (IOU) with ground truth is larger than some overlap threshold σ .

6.1. Results on THUMOS 2014

THUMOS '14 includes 20 sports action classes for temporal detection, with over 13K short clips and background videos for training, 1010 untrimmed validation videos, and 1574 untrimmed videos for testing. In our experiments, we use the training clips, background, and validation videos for training, and report results on the untrimmed test videos. During training, we crop each validation video to short 800-frame clips which contain one single action instance. We augment the training dataset by splicing action instances from the training and validation clips with background videos and validation videos in which no instances of the 20 classes appear. In total, we generate 42000 action clips for training. We choose hyperparameters based on results on a withheld subset of the validation videos. We train Spatial-CNN and Motion-CNN for 16K and 20K iterations, respectively. We then finetune the two-stream network for 2K additional iterations. At test time, we downsample all videos to 5fps, and filter out videos that are unlikely to contain any of the 20 action classes. We do this by averaging their frame-level class scores from action recognition models [27] finetuned on THUMOS'14. In Figure 4, we show example detections on the THUMOS'14 test set.

We report results at varying overlap thresholds in Table 1 and compare with existing systems. Our model outperforms state-of-the-art when the overlap threshold σ is 0.1, 0.2 and 0.3, and achieves competitive results for 0.4 and 0.5. This indicates that our system can distinguish action instances from background frames even when precise localization is difficult. Additionally, we provide per-class average precision results in Figure 5. Our system achieves the best performance on 5 of the 20 actions. For a few actions, namely *Billiards*, *Cricket Shot*, *Tennis Swing* and *Volleyball Spiking*, we get a relatively low average precision. This could be due

Comparison with State-of-the-Art

overlap threshold σ	0.1	0.2	0.3	0.4	0.5
Karaman <i>et al.</i> [10]	1.5	0.9	0.5	0.3	0.2
Wang <i>et al.</i> [26]	19.2	17.8	14.6	12.1	8.5
Oneata <i>et al.</i> [11]	39.8	36.2	28.8	21.8	15.0
Shou <i>et al.</i> [15]	47.7	43.5	36.3	28.7	19.0
Yeung <i>et al.</i> [29]	48.9	44.0	36.0	26.4	17.1
Richard <i>et al.</i> [12]	39.7	35.7	30.0	23.2	15.2
Ours (full)	51.0	45.2	36.5	27.8	17.8

Table 1: The mean average precision (mAP) of different methods for varying overlap thresholds. Our system achieves state-of-the-art performance for $\sigma = 0.1, 0.2, 0.3$.

Ablation Study

overlap threshold σ	0.1	0.2	0.3	0.4	0.5
Baseline	18.5	10.2	4.5	1.8	0.2
w/o <i>cls</i> + <i>sme</i>	40.5	28.8	23.2	16.4	13.2
w/o <i>cls</i>	42.5	32.6	27.8	19.6	15.7
w/o <i>sme</i>	48.0	42.2	33.0	24.8	16.2
w/o <i>prior</i>	50.7	45.0	36.2	27.4	17.5
Ours (full)	51.0	45.2	36.5	27.8	17.8

Separate networks

<i>spatial</i>	46.2	40.3	31.5	23.2	16.0
<i>motion</i>	47.6	44.0	35.6	25.8	16.9
<i>late fusion</i>	46.0	43.2	32.8	24.0	14.5
Ours (full)	51.0	45.2	36.5	27.8	17.8

Table 2: Ablation experiments for the structured objective (top) and the two-stream architecture (bottom). The full model outperforms all other configurations.

to two main reasons: (1) these action instances are short and thus the action scores are relatively noisy compared to longer actions and (2) these actions often occur in rapid succession, making it easy to merge adjacent action instances.

Ablation Study In order to show the contribution of each component in our system, we experiment with eight variants of the full pipeline. The results are reported in Table 2.

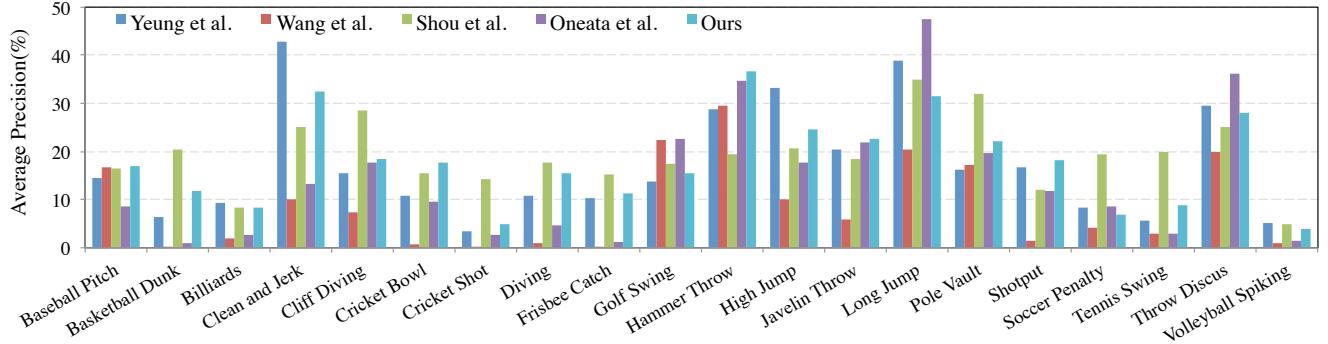


Figure 5: Per-class average precision on THUMOS ’14 at overlap threshold $\sigma = 0.5$.



Figure 6: Example detections from our full system compared with our system trained without the classification loss \mathcal{L}_{cls} .

In *baseline*, we do not train using a structured loss function during training. Instead, we train our model to perform mutually exclusive framewise classification, and subtract the mean confidence to form signed confidence scores, and perform localization using SMS. In *w/o cls*, we drop the structured localization loss \mathcal{L}_{cls} during training. In *w/o sme*, we do not model start-middle-end temporal evolution, and each action score is computed as the average of frame-level action classification scores. In *w/o cls + sme*, we use drop \mathcal{L}_{cls} and also do not model temporal evolution. Dropping each of these components results in a significant drop in performance, indicating that both temporal evolution and the structured classification loss are important to facilitate training and accurate localization. In *w/o prior*, we drop the action duration priors, which results in a small drop in performance. In Figure 6, we give examples of detections produced when the \mathcal{L}_{cls} loss is dropped, localization becomes less precise. We also perform a separate ablation study on the components of the two-stream architecture. We evaluate using each stream individually (*spatial* and *motion*), as well as simply averaging the two streams rather than fine-tuning jointly (*late fusion*). We find that the full model outperforms

late fusion, suggesting the importance of joint training of the two streams. The *late fusion* does not outperform separate networks, suggesting an incompatibility of confidence scores from the two networks.

6.2. Conclusions

We present a framework for end-to-end training of temporal localization that takes into account the *temporal evolution* of each action. We frame localization as a *Structured Maximal Sum* problem, and provide efficient algorithms for training and detection in this framework. We show that modeling temporal evolution improves performance, and demonstrate that our system achieves competitive performance on the THUMOS ’14 benchmark.

Acknowledgements

This work is partially supported by a University of Michigan graduate fellowship, the Natural Science Foundation of China under Grant No. 61672273, No. 61272218, and No. 61321491, and the Science Foundation for Distinguished Young Scholars of Jiangsu under Grant No. BK20160021.

References

- [1] S. E. Bae and T. Takaoka. Algorithms for the problem of K maximum sums and a VLSI algorithm for the K maximum subarrays problem. In *International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, pages 247–253, 2004.
- [2] J. L. Bentley. *Programming pearls*. Addison-Wesley, 1986.
- [3] G. S. Brodal and A. G. Jørgensen. A linear time algorithm for the k maximal sums problem. In *International Symposium on Mathematical Foundations of Computer Science*, pages 442–453. Springer, 2007.
- [4] A. Gaidon, Z. Harchaoui, and C. Schmid. Temporal localization of actions with actoms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2782–2795, 2013.
- [5] C. Gentile and M. K. Warmuth. Linear hinge loss and average margin. In *Advances in Neural Information Processing Systems (NIPS)*, pages 225–231, 1998.
- [6] A. Gorban, H. Idrees, Y.-G. Jiang, A. Roshan Zamir, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. <http://crcv.ucf.edu/THUMOS14/>, 2014.
- [7] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Computer Vision and Pattern Recognition (CVPR)*, pages 961–970, 2015.
- [8] M. Hoai and F. De la Torre. Max-margin early event detectors. *International Journal of Computer Vision (IJCV)*, 107(2):191–202, 2014.
- [9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia (MM)*, pages 675–678, 2014.
- [10] S. Karaman, L. Seidenari, and A. D. Bimbo. Fast saliency-based pooling of fisher encoded dense trajectories. *ECCV THUMOS Workshop*, 2014.
- [11] D. Oneata, J. J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1817–1824, 2013.
- [12] A. Richard and J. Gall. Temporal action detection using a statistical language model. *CVPR*, 2016.
- [13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [14] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: primal estimated sub-gradient solver for SVM. *Math. Program.*, 127(1):3–30, 2011.
- [15] Z. Shou, D. Wang, and S. Chang. Action temporal localization in untrimmed videos via multi-stage cnns. *CoRR*, abs/1601.02129, 2016.
- [16] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014.
- [17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [18] B. Singh and M. Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. *CVPR*, 2016.
- [19] G. Singh and F. Cuzzolin. Untrimmed video classification for activity detection: submission to activitynet challenge. *arXiv preprint arXiv:1607.01979*, 2016.
- [20] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.
- [21] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *ACM Conference on Multimedia MM ’15*, pages 371–380, 2015.
- [22] K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1250–1257. IEEE, 2012.
- [23] D. Tran and J. Yuan. Optimal spatio-temporal path discovery for video event detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3321–3328, 2011.
- [24] D. Tran and J. Yuan. Max-margin structured output regression for spatio-temporal action localization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 359–367, 2012.
- [25] H. Wang and C. Schmid. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 3551–3558, 2013.
- [26] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. *THUMOS14 Action Recognition Challenge*, 1:2, 2014.
- [27] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *CoRR*, abs/1507.02159, 2015.
- [28] P. Wei, N. Zheng, Y. Zhao, and S.-C. Zhu. Concurrent action detection with structural prediction. In *International Conference on Computer Vision (ICCV)*, pages 3136–3143, 2013.
- [29] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. *CVPR*, 2016.
- [30] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv- L^1 optical flow. In *German Association for Pattern Recognition Symposium (DAGM)*, pages 214–223, 2007.